Myers, C. J., Beal, J., Gorochowski, T. E., Kuwahara, H., Madsen, C., McLaughlin, J. A., Misirli, G., Nguyen, T., Oberortner, E., Samineni, M., Wipat, A., Zhang, M., & Zundel, Z. (2017). A standard-enabled workflow for synthetic biology. *Biochemical Society Transactions*, *45*(3), 793-803. https://doi.org/10.1042/BST20160347

A Standard-Enabled Workflow for Synthetic Biology

Chris J. Myers,[*,†] Jacob Beal,[*,‡] Thomas E. Gorochowski,[*,¶] Hiroyuki Kuwahara,[§] Curtis Madsen,[*,‖] James Alastair McLaughlin,[*,⊥] Göksel Mısırlı,[*,⊥] Tramy Nguyen,[*,†] Ernst Oberortner,[*,#] Meher Samineni,[*,†] Anil Wipat,[*,⊥] Michael Zhang,[*,†] and Zach Zundel[*,†]

†*Department of Electrical and Computer Engineering, University of Utah, 50 S. Central Campus Drive, Rm. 2110, Salt Lake City, UT, 84112, USA*

‡*Information and Knowledge Technologies, Raytheon BBN Technologies, 10 Moulton Street, Cambridge, MA 02138*

¶*BrisSynBio, University of Bristol, Life Sciences Building, Tyndall Avenue, Bristol BS8 1TQ, UK*

§*King Abdullah University of Science and Technology (KAUST), Computational Bioscience Research Center (CBRC), Thuwal, 23955 Saudi Arabia*

‖*Department of Electrical and Computer Engineering, Boston University, 8 Saint Marys Street, Office #324, Boston, MA 02215, United States*

⊥*School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU, UK*

#*DOE Joint Genome Institute, 2800 Mitchell Drive, Walnut Creek, CA, 94598, USA*

E-mail: myers@ece.utah.edu; jakebeal@ieee.org; thomas.gorochowski@bristol.ac.uk; ckmadsen@bu.edu; j.a.mclaughlin@ncl.ac.uk; gokselmisirli@gmail.com; tramy.nguyen@utah.edu; eoberortner@lbl.gov; m.samineni@utah.edu; anil.wipat@ncl.ac.uk; michael.zhang@utah.edu; zach.zundel@utah.edu

Abstract

A synthetic biology workflow is composed of data repositories that provide information about genetic parts, sequence-level design tools to compose these parts into circuits, visualization tools to depict these designs, genetic design tools to select parts to create systems, and modeling and simulation tools to evaluate alternative design choices. Data standards enable the ready exchange of information within such a workflow, allowing repositories and tools to be connected from a diversity of sources. This paper describes one such workflow that utilizes, among others, the Synthetic Biology Open Language (SBOL) to describe genetic designs, the Systems Biology Markup Language (SBML) to model these designs, and SBOL Visual to visualize these designs. We describe how a standard-enabled workflow can be used to produce types of design information, including multiple repositories and software tools exchanging information using a variety of data standards. Recently, the ACS Synthetic Biology journal has recommended the use of SBOL in their publications.

Introduction

Reproducibility is a critical and growing issue in synthetic biology.[1] Substantial effort is often required to design a new biological system, with input from many researchers with different backgrounds, including biology, mathematics, computer science, physics, chemistry, etc. Extracting information in order to reuse or build upon the contributions made by these researchers, however, is often extremely challenging. At present, information about genetic circuit design is often incomplete or buried in textual descriptions. Even scientific publications often fail to fully convey this information: designs are often available only as visual depictions that provide abstract representations or as unannotated sequences, and frequently some of the genes or gene products are not even captured, making it nearly impossible to reuse these designs. Capturing DNA sequences is key as a first step, but this information may also not be available, may require lengthy and error-prone manual lookups

based on gene identifiers, or may only be derivable by search and extraction of the partial sequences given in forward and reverse primers. Even then, deriving exact sequences of designs may be impossible when full information about the final design, such as its exact assembly process, cloning strategy, or the spacer sequences between constituent genes and their components, are not clearly specified.

Further complicating matters, experimental measurements may vary between different labs due to the differences in sequences, chassis organisms or the lack of information about experimental conditions. Even single nucleotide differences between sequences in the design itself or the host chassis can significantly change the functionality of genetic circuits. Notably, modifications in non-coding sequences can strongly affect the rates of transcription and translation processes, resulting in unexpected behaviors.[2–5] As the scale and the complexity of designs increases, these problems bring more challenges.

As synthetic biology continues to develop as an engineering discipline, practitioners are grappling with these problems and adopting the same sort of strategies that enable management of complexity in every mature engineering discipline, such as standardization, abstraction, modularity, and automation. Applications are created through design-build-test cycles and automation is key to achieve faster cycles for commercialization. There are already a wide variety of available computational tools that can be used in different stages of design, manufacturing, testing, and analysis. Often, tools are specialized in performing specific functions, and synthetic biology engineers need to flexibly coordinate the operation of these tools in complex design workflows. As a result, computational access to and exchange of information without any loss is crucial. Finally, the use of standards to capture design information also enhances reproducibility and reusability,[6] effectively allowing the products of one workflow to be consumed by other workflows. Computational access particularly facilitates the storage and retrieval of these designs, making them ever more accessible. Practitioners can therefore more easily find designs that are created by other practitioners in a timely manner, make modifications or reuse them, and electronically share their new designs and data.

Synthetic Biology Open Language (SBOL)

The *Synthetic Biology Open Language* (SBOL) is one of the key technologies that can support the emerging standards-driven approach to synthetic biology engineering workflows. SBOL is a free and open community standard for the description and exchange of biological designs, supported by a diverse international community of researchers. This standard provides a "common core" set of relatively abstract representations of biological structure, function, and sequence, with a focus on abstraction and composition, and is broadly applicable across a wide range of workflow elements. Critically, SBOL also supports machine-interpretable links between this shared core and more specialized representations, such as numerical models, protocol automation scripts, LIMS tracking, and measurement data, allowing SBOL to serve as a "hub" for linking together a wide range of more specialized tools and processes without loss of information as shown in Figure 1.

The development of SBOL was motivated by the shortcomings of prior standards, such as FASTA[7] and GenBank,[8] with respect to describing the engineering of biological systems. These prior standards focus on the recording and annotation of natural nucleic acid or protein sequence data, which has different challenges and requirements than the engineering of novel human-designed biological constructs. For example, the description of engineered systems requires the representation of the abstraction and composition of (at least partially) modular components. To serve these needs, in 2008, the SBOL community developed first an initial draft standard called PoBol,[9] which evolved into first the SBOL 1 standard,[10,11] focusing on the genetic structure of engineered DNA sequences. SBOL 1 recently evolved into the SBOL 2 standard,[12,13] which represents both structure and function of genetic designs as depicted in Figure 2.

Complementary to this data model, the SBOL visual standard provides a common visual language for communication about engineering biological constructs, much as diagram languages for electrical engineering[14,15] and architecture[16,17] do in those fields. SBOL Visual (SBOLv)[18,19] enables diagrams for SBOL 1 constructs, and is in the process of being

extended and integrated with *Systems Biology Graphical Notation* (SBGN)[20] to support the functional representations of SBOL 2 as well. SBOL visual is formally related to the SBOL data representation by means of the *Sequence Ontology* (SO),[21] which is used by the SBOL data model to designate the roles of components as shown in Figure 3. Namely, each glyph in SBOL visual is mapped to one or more ontology terms, enabling automatic computational mapping from SBOL data models to diagrams, by selecting for each component the most specific glyph whose term covers the component's role or roles and organizing these glyphs according to the sequence and order relationships specified in the data model.

Supporting Reproduction and Reuse with SBOL

In order to support effective reproduction and reuse, practitioners must not only have the capability to represent information about engineered biological organisms, but must also use those capabilities to encode enough information of the right types to enable others to reproduce or build upon their results. In mature engineering fields, this typically takes the form of formalized datasheets, such as the component datasheets used in electronics or Materials Safety Data Sheets used in chemistry. Although biological organism engineering aspires to this level of rigor (e.g.,[22]), in practice the field has not yet attained that level of maturity.[23] In other areas of biology, the challenges of reproduction and reuse are addressed with a variety of *minimum information standards*,[24] which aim to at least ensure that enough information on protocol and context is included that a practitioner can determine whether an attempt to reproduce or reuse works as expected. For example, MIAME establishes minimum information standards for reporting on micro-array experiments,[25] and MIFlowCyt establishes minimum information standards for reporting on flow cytometry experiments.[26] By making it easier to compare the products of different efforts, such minimum information standards have significantly improved data quality and accelerated discovery in the areas in which they have been established.

Similarly, reproduction and reuse of genetic constructs should be able to be accelerated by establishing a reporting standard for the minimum information about a genetic construct. Such minimum information about a genetic construct or collection of constructs needs to include at least the following:

- The full sequence of all of the "base" components used in a genetic construct. For example, a library made by combining pairs of promoters and coding sequences would need to include the full sequence of every promoter and every coding sequence.

- Information sufficient to unambiguously determine the sequence of every complete construct. For example, the promoter/coding-sequence library would record all combinations made, but not necessarily the sequence of each combination, if that can be determined from the combination and the sequences of the individual components.

- Identification of the role played by each significant designed feature. For example, explicitly recording that each promoter is, in fact, a promoter.

- Identification of identities between construct components, such as by the composition of sub-components. For example, it should be easy to tell if two promoter/coding-sequence constructs share the same promoter.

- The assembly method used, if any, for composing smaller components into larger components, and any effects this is expected to have on the resulting sequence.

- Any required additional modifications of the base sequence, such as methylation.

- The vector or integration point used for transformation of the host organism. For example, a plasmid used to deliver a construct to bacteria, or the location targeted for CRISPR-based integration into a chromosome.

- An unambiguous identification of the host organism for the construct, sufficient for determining genome and other relevant features.

The core representations of SBOL readily support most of this information, while the remainder can be linked to SBOL representations via the annotation mechanisms provided by SBOL, and an effort is ongoing within the SBOL community to formalize these recommendations.

Already, journals have shown interest in using SBOL to improve the ability of readers to reproduce and reuse elements of the papers they publish. In 2016, ACS Synthetic Biology became the first journal to formally embrace SBOL as a means of enhancing reproduction and reuse of synthetic biology research,[27] with a workflow including validation and review of submitted designs and their deposit into a design repository linked with the paper and with interfaces for access by both humans and genetic design automation tooling as shown in Figure 4. As minimum information standards are established and adopted, they can integrate with such workflows in order to improve the ability of the research community to reproduce and to build upon one another's results. In parallel, we may expect such standards to provide a basis for development of a wide variety of new capabilities, services, and business models in the industrial community, much as shared standards have already done in other communities, such as software, electronics, and mechanical systems.

Software Support for SBOL

Crucial to the success of SBOL is software infrastructure to support developers' integration of this standard within their tools. In particular, several libraries have been developed that provide access to the data model through an *application programming interface* (API). These libraries also permit both the serialization of data objects into RDF/XML and the parsing of SBOL files into SBOL data objects for ease of interaction and manipulation within software tools. There are currently four main libraries maintained in loose federation by members of the SBOL community: libSBOLj (Java),[28] libSBOL (C/C++), pySBOL (Python), and sboljs (JavaScript). The Java library provides methods for converting to/from FASTA, GenBank,

SBOL 1, and SBOL 2, as well as methods to check an SBOL document against the validation rules outlined in the SBOL specification.[29] The SBOL Validator/Converter provides a web service that can be leveraged by non-Java software to access these functionalities.

Leveraging these libraries, a number of software applications that support the SBOL standard have been developed as shown in Table 1. These tools can be loosely divided into data repositories for storing genetic design information, sequence editors, visualization tools, genetic design compilers, and modeling and simulation tools. Many of these applications actually cover more than one of these functions. While most of these tools support either SBOL 1 or SBOLv, an increasing number of tools supporting SBOL 2 are being released. The rest of this section provides a brief description of some of these software tools. More detailed descriptions can be found in the supplemental material.

Several data repositories have been developed that can store genetic design information using the SBOL data standard. ICE[30] is an open-source software tool that provides a Web-based platform to register and manage DNA parts, and an instance of this platform is used as the ACS Synthetic Biology Registry.[27] SynBioHub is an open-source repository built upon the SBOL Stack[31] RDF (resource description framework) database back-end, and it provides both a user-friendly web-based front-end and programmatic access via either libSBOLj or a RESTful API. SBOLme is a Web-based open-access repository that has recently been developed to promote the use of the SBOL for metabolic engineering applications.[32] The first release of SBOLme contains annotated SBOL parts of 28,437 chemical compounds, 6,883 enzyme classes, 9,909 metabolic reactions, and 3,173,238 proteins from 3,908 different organisms. Finally, the Virtual Parts Repository supports CAD tools by providing readily accessible modular and reusable models of biological components that can be individually joined together for simulation.[33]

Sequence editors are software tools for the design of DNA, RNA, and/or protein sequences. The task of designing sequences incorporates the manipulation, composition, and annotation of sequences. There are many tools developed or being developed with these

functions, we highlight here a few with the best SBOL support, while more are described in the supplemental material. Eugene enables the specification of rules in order to automatically enumerate composited designs based on biological knowledge.[34] The Joint BioEnergy Institute (JBEI) develops DeviceEditor[35] to visually design combinatorial DNA constructs based on part types (e.g., promoter, CDS, terminator), VectorEditor for a graphical preview of the design, and j5 for DNA assembly design automation. SBOLDesigner is a modular sequence design tool that combines the SBOL 2 data model with SBOLv symbols to construct genetic designs hierarchically using parts fetched from SBOL data repositories.[36] The Build-Optimization Software Tools (BOOST)[37] enable the design of DNA sequences in order to maximize the success rate of their synthesis via codon optimization, verification of sequence constraints, and decomposition into synthesizable blocks.

SBOLv defines a set of agreed symbols to denote commonly used genetic elements and best-practices for how biological designs should be visualized. Many point-and-click genetic design tools have adopted these symbols (see Table 1) and several dedicated pieces of software are now available to simplify the process of generating compliant diagrams. One of the first tools to help automate the production of standardized SBOLv diagrams was Pigeon,[38] which converts a textual input description of a genetic construct into a diagram where each part is represented by its associated SBOLv symbol. Highly customized SBOLv diagrams can be created by using the DNAplotlib computational toolkit.[39] VisBOL is a Web-based tool that in addition to supporting the Pigeon syntax can also convert directly from an SBOL 2 document into SBOLv symbols.[40] Finally, SBOL visual symbols have been adopted into the widely used general graph visualization toolkit, Graphviz.

Genetic circuit design involves constructing biological systems that implement logical functions similar to those found in electronic circuits. Circuit designers usually build circuits by connecting parts or modules found in a library to form larger and more complex constructs. Many tools have been developed that attempt to assist engineers in genetic circuit design. Proto BioCompiler takes in specifications of computations, transforms them into

a data-flow representation of the computation to be carried out by the biological organism, then selects parts from a genetic library, and finally optimizes the circuit design.[41] iBioSim adapts a graph-based technology mapping procedure from digital electronic circuit design to map a specified genetic regulatory model into a network of genetic gates specified using SBOL.[42] Finally, Cello provides a platform where users can describe the desired function of their genetic circuit using Verilog, a hardware description language commonly used to specify electronic circuits, and then translate it into a directed acyclic graph of connected 2-input NOR and NOT gates implementing the logic.[43]

Finally, SBOL allows for the association of genetic circuit designs with computational models. The most commonly used data standard for models of biological systems is the *Systems Biology Markup Language* (SBML).[44] SBML models can be analyzed using a large selection of different analysis methods including deterministic and stochastic simulation,[45] flux balance analysis,[46] and stochastic model checking.[47] To facilitate the construction of SBML models, a converter from SBOL to SBML has been developed.[48] It is also possible to begin with an SBML model annotated with SBOL[49] and produce an SBOL description for the genetic design.[48] Given an SBML model for a genetic design, it is then possible to analyze this model using a variety of SBML modeling tools including those optimized for genetic circuit design, such as iBioSim,[50,51] Tellurium,[52] and TinkerCell.[53]

A Standard-Enabled Workflow for Synthetic Biology

A key design principle in the development of SBOL is that it would not attempt to cover all aspects of genetic design, but rather it would leverage existing standards whenever possible. A key example of this is the use of SBML for modeling. To pursue this goal, SBOL recently joined the COMBINE (*COmputational Modeling in BIology NEtwork*) community of standards.[6] COMBINE is an open community initiative to coordinate the development of standards and formats for systems and synthetic biology. Figure 5 depicts a complete

synthetic biology computational design workflow that leverages COMBINE standards. This workflow assumes that data required for design must come from a variety of data repositories. While some are SBOL repositories, others store their information in other formats such as GenBank or BioPAX ,[54] another COMBINE standard. Converters can be utilized to translate this knowledge into SBOL to be utilized during sequence design using any of the sequence editors and visualization tools described earlier. Next, genetic modeling, analysis, and design tools can be utilized to construct and evaluate complete genetic designs. These models would be constructed using a COMBINE modeling language such as SBML or CellML,[55] and their analyses should be encoded using the *Simulation Experiment Description Markup Language* (SED-ML).[56] Next, SBOLv only represents DNA constructs, so a visualization standard such as SBGN could be leveraged to represent the biochemical aspects of the design. Finally, each of these files can be packaged together, shared, and distributed using a COMBINE Archive.[57] Throughout, the data conversions required by this standard-enabled workflow are enabled by the use of common ontologies, such as the BioPAX Ontology,[54] the Sequence Ontology (SO),[58] and the Systems Biology Ontology (SBO)[59] with URIs taken from identifiers.org,[60] whenever possible.

Conclusion

Standards are an important enabler for data sharing and reproducibility in synthetic biology. Collaborations within the COMBINE community are essential to create new workflows enabled by these standards. The ultimate goal of these collaborations is a complete standard-enabled workflow for synthetic biology. For more information about SBOL, please see our website: `http://www.sbolstandard.org/`, and YouTube channel that includes several demonstrations of the standard-enabled workflow that we are developing.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants CCF-1218095 and DBI-135604. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. T.E.G. is supported by BrisSynBio, a BBSRC/EPSRC Synthetic Biology Research Centre (BB/L01386X/1); G.M. and A.W. have been supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J02175X/1. J.A.M is supported by FUJIFILM DioSynth Biotechnologies. J.S.B. is supported in part by the National Science Foundation Expeditions in Computing Program Award #1522074 as part of the Living Computing Project. E.O. is supported under Contract No. DE-AC02-05CH11231 by the U.S. Department of Energy Joint Genome Institute, a DOE Office of Science User Facility. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

References

(1) Peccoud, J.; Anderson, J. C.; Chandran, D.; Densmore, D.; Galdzicki, M.; Lux, M. W.; Rodriguez, C. A.; Stan, G.-B.; Sauro, H. M. Essential information for synthetic DNA sequences. *Nat. Biotechnol.* **2011**, *29*, 22.

(2) Cardinale, S.; Arkin, A. P. Contextualizing context for synthetic biology–identifying causes of failure of synthetic biological systems. *Biotechnol J* **2012**, *7*, 856–66.

(3) Davis, J. H.; Rubin, A. J.; Sauer, R. T. Design, construction and characterization of a set of insulated bacterial promoters. *Nucleic Acids Res* **2011**, *39*, 1131–41.

(4) Meng, W.; Belyaeva, T.; Savery, N. J.; Busby, S. J.; Ross, W. E.; Gaal, T.; Gourse, R. L.; Thomas, M. S. UP element-dependent transcription at the Escherichia coli rrnB P1

Table 1: A partial list of software supporting SBOL. An up-to-date list is maintained on `http://sbolstandard.org`. The function column indicates if the tool is a (R)epository, (S)equence design tool, (G)enetic circuit design tool, (M)odeling and simulation tool, or a (V)isualization tool. The SBOL column indicates if it supports SBOL(1), (2), or (v)isual.

	Function					**SBOL**			
Name	R	S	V	G	M	1	2	v	**URL**
Benchling[35]		•				•			benchling.com
BOOST[37]		•				•	•		boost.jgi.doe.go
Cello[61]				•			•		cellocad.org
DeviceEditor[35]		•	•			•		•	j5.jbei.org
DNAPlotLib[62]			•			•		•	dnaplotlib.org
Eugene[34]		•				•		•	http://www.eugenecad.org
Finch		•	•	•			•	•	synbiotools.org
GenoCAD[63]		•	•					•	www.genocad.com
GeneGenie		•				•			gene-genie.org
Graphviz			•					•	www.graphviz.org
ICE[30]	•		•			•	•	•	public-registry.jbei.org
iBioSim[64]		•	•	•	•	•	•	•	www.async.ece.utah.edu/ibiosim
j5[65]		•							j5.jbei.org
MoSeC[66]		•			•	•			ico2s.org/software/mosec.html
Pigeon[67]			•					•	pigeoncad.org
Pinecone		•						•	serotiny.bio
Pool Designer[68]		•				•	•		github.com/CIDARLAB/poolDesigner
Proto BioCompiler[41]			•	•		•		•	synbiotools.bbn.com
SBOLDesigner[36]		•	•			•	•	•	www.async.ece.utah.edu/SBOLDesigner
SBOLme[32]	•						•		www.cbrc.kaust.edu.sa/sbolme
ShortBol[69]		•		•			•		shortbol.ico2s.org/sandbox.html
SynBioHub[31]	•		•			•	•	•	synbiohub.org
Tellurium[52]					•		•		tellurium.analogmachine.org
TeselaGen		•	•			•		•	www.teselagen.com
TinkerCell[53]			•	•	•	•		•	www.tinkercell.com
VisBOL[70]			•				•	•	visbol.org
VirtualParts[33]	•				•		•		www.virtualparts.org

promoter: positional requirements and role of the RNA polymerase α subunit linker. *Nucleic acids research* **2001**, *29*, 4166–4178.

(5) Iverson, S. V.; Haddock, T. L.; Beal, J.; Densmore, D. M. CIDAR MoClo: Improved MoClo Assembly Standard and New E. coli Part Library Enable Rapid Combinatorial Design for Synthetic and Traditional Biology. *ACS Synth Biol* **2016**, *5*, 99–103.

(6) Hucka, M.; Nickerson, D. P.; Bader, G. D.; Bergmann, F. T.; Cooper, J.; Demir, E.; Garny, A.; Golebiewski, M.; Myers, C. J.; Schreiber, F.; Waltemath, D.; Le Novère, N. Promoting Coordinated Development of Community-Based Information Standards for Modeling in Biology: The COMBINE Initiative. *Frontiers in Bioengineering and Biotechnology* **2015**, *3*, 19.

(7) Pearson, W. R.; Lipman, D. J. Improved tools for biological sequence comparison. *P. Natl. Acad. Sci. USA* **1988**, *85*, 2444–2448.

(8) Bilofsky, H. S.; Christian, B. The GenBank genetic sequence data bank. *Nucleic Acids Res.* **1988**, *16*, 1861–1863.

(9) Galdzicki, M.; Chandran, D.; Nielsen, A.; Morrison, J.; Cowell, M.; Grünberg, R.; Sleight, S.; Sauro, H. *BBF RFC 31: Provisional BioBrick Language (PoBoL)*; 2009.

(10) Galdzicki, M. et al. Synthetic Biology Open Language (SBOL) Version 1.1.0. BBF RFC 87, DOI: 1721.1/73909.

(11) Galdzicki, M. et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol.* **2014**, *32*, 545–550.

(12) Bartley, B.; Beal, J.; Clancy, K.; Misirli, G.; Roehner, N.; Oberortner, E.; Pocock, M.; Bissell, M.; Madsen, C.; Nguyen, T.; Zhang, Z.; Gennari, J.; Myers, C.; Wipat, A.;

Sauro, H. Synthetic Biology Open Language (SBOL) Version 2.0.0. *Journal of Integrative Bioinformatics* **2015**, *12*, 272.

(13) Roehner, N. et al. Sharing structure and function in biological design with SBOL 2.0. *ACS Synthetic Biology* **2016**, *5*, 498–506.

(14) IEEE, IEEE Graphic Symbols for Logic Functions (Includes IEEE Std 91A-1991 Supplement, and IEEE Std 91-1984). IEEE Std. 91a-1991, 1991.

(15) IEEE, IEEE Standard American National Standard Canadian Standard Graphic Symbols for Electrical and Electronics Diagrams (Including Reference Designation Letters). IEEE Std. 315-1975 (Reaffirmed 1993), 1993.

(16) Schley, M.; Buday, R.; Sanders, K.; Smith, D. *AIA CAD layer guidelines*; The American Institute of Architects Press: Washington, DC, 1997.

(17) British Standards Institution, Collaborative production of architectural, engineering and construction information. BS 1192:2007, 2007.

(18) Quinn, J.; Beal, J.; Bhatia, S.; Cai, P.; Chen, J.; Clancy, K.; Hillson, N.; Galdzicki, M.; Maheshwari, A.; Pocock, M.; Rodriguez, C.; Stan, G.-B.; Endy, D. *Synthetic biology open language visual (SBOL visual), version 1.0.0*; 2013; BioBricks Foundation Request for Comments (BBF RFC) #93.

(19) Quinn, J. et al. SBOL Visual: A Graphical Language for Genetic Designs. *PLoS Biol* **2015**, *13*, e1002310.

(20) Le Novere, N. et al. The systems biology graphical notation. *Nature biotechnology* **2009**, *27*, 735–741.

(21) Eilbeck, K.; Lewis, S. E.; Mungall, C. J.; Yandell, M.; Stein, L.; Durbin, R.; Ashburner, M. The Sequence Ontology: a tool for the unification of genome annotations. *Genome biology* **2005**, *6*, R44.

(22) Canton, B.; Labno, A.; Endy, D. Refinement and standardization of synthetic biological parts and devices. *Nature Biotechnology* **2008**, *26*, 787–93.

(23) Kwok, R. Five hard truths for synthetic biology. *Nature* **2010**, *463*, 288–90.

(24) Taylor, C. F. et al. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. *Nature biotechnology* **2008**, *26*, 889–896.

(25) Brazma, A. et al. Minimum information about a microarray experiment (MIAME)?toward standards for microarray data. *Nature genetics* **2001**, *29*, 365–371.

(26) Lee, J. A. et al. MIFlowCyt: the minimum information about a Flow Cytometry Experiment. *Cytometry Part A* **2008**, *73*, 926–930.

(27) Hillson, N.; Plahar, H.; Beal, J.; Prithviraj, R. Improving Synthetic Biology Communication: Recommended Practices for Visual Depiction and Digital Submission of Genetic Designs. *ACS synthetic biology* **2016**, *5*, 449–451.

(28) Zhang, Z.; Nguyen, T.; Roehner, N.; Misirli, G.; Pocock, M.; Oberortner, E.; Samineni, M.; Zundel, Z.; Beal, J.; Clancy, K.; Wipat, A.; Myers, C. libSBOLj 2.0: a java library to support SBOL 2.0. *IEEE Life Sciences Letters* **2015**, *1*, 34–37.

(29) Zundel, Z.; Samineni, M.; Zhang, Z.; Myers, C. A Validator and Converter for the Synthetic Biology Open Language. *ACS Synthetic Biology* **2016**,

(30) Ham, T. S.; Dmytriv, Z.; Plahar, H.; Chen, J.; Hillson, N. J.; Keasling, J. D. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res.* **doi: 10.1093/nar/gks531, 2012**, *40*.

(31) Madsen, C.; McLaughlin, J.; Misirli, G.; Pocock, M.; Flanagan, K.; Hallinan, J.; Wipat, A. The SBOL Stack: A Platform for Storing, Publishing, and Sharing Synthetic Biology Designs. *ACS synthetic biology* **2016**, *5*, 487–497.

(32) Kuwahara, H.; Cui, X.; Umarov, R.; Grünberg, R.; Myers, C. J.; Gao, X. SBOLme: a Repository of SBOL Parts for Metabolic Engineering. *ACS Synth Biol* **2017**,

(33) Misirli, G.; Hallinan, J.; Wipat, A. Composable Modular Models for Synthetic Biology. *ACM J. Emerg. Technol. Comput. Syst.* **2014**,

(34) Bilitchenko, L.; Liu, A.; Cheung, S.; Weeding, E.; Xia, B.; Leguia, M.; Anderson, J. C.; Densmore, D. Eugene–a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS ONE* **doi: 10.1371/journal.pone.0018882, 2011**, *6*, e18882.

(35) Chen, J.; Densmore, D.; Ham, T. S.; Keasling, J. D.; Hillson, N. J. DeviceEditor visual biological CAD canvas. *Journal of Biological Engineering* **2012**, *6*, 1.

(36) Zhang, M.; McLaughlin, J.; Wipat, A.; Myers, C. SBOLDesigner 2: An Intuitive Genetic Circuit Design Tool. *(forthcoming)* **2017**,

(37) Oberortner, E.; Cheng, J.-F.; Hillson, N. J.; Deutsch, S. Streamlining the Design-to-Build Transition with Build-Optimization Software Tools. *ACS Synthetic Biology* **0**, *0*, null, PMID: 28004921.

(38) Bhatia, S.; Densmore, D. Pigeon: A Design Visualizer for Synthetic Biology. *ACS Synthetic Biology* **2013**, *2*, 348–350.

(39) Der, B. S.; Glassey, E.; Bartley, B. A.; Enghuus, C.; Goodman, D. B.; Gordon, D. B.; Voigt, C. A.; Gorochowski, T. E. DNAplotlib: Programmable Visualization of Genetic Designs and Associated Data. *ACS Synthetic Biology* **2016**,

(40) McLaughlin, J.; Pocock, M.; Misirli, G.; Madsen, C.; Wipat, A. VisBOL: Web-Based Tools for Synthetic Biology Design Visualization. *ACS Synthetic Biology* **2016**, *5*, 874–876.

(41) Beal, J.; Lu, T.; Weiss, R. Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PloS one* **2011**, *6*, e22490.

(42) Roehner, N.; Myers, C. J. Directed acyclic graph-based technology mapping of genetic circuit models. *ACS Synth. Biol.* **2014**, *3*, 543–555.

(43) Nielsen, A. A. K.; Der, B. S.; Shin, J.; Vaidyanathan, P.; Paralanov, V.; Strychalski, E. A.; Ross, D.; Densmore, D.; Voigt, C. A. Genetic circuit design automation. *Science* **2016**, *352*.

(44) Hucka, M. et al. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **2003**, *19*, 524–531.

(45) Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry* **1977**, *81*, 2340–2361.

(46) Edwards, J.; Covert, M.; Palsson, B. Metabolic modelling of microbes: the flux-balance approach. *Environmental Microbiology* **2002**, *4*, 133–140.

(47) Madsen, C.; Zhang, Z.; Roehner, N.; Winstead, C.; Myers, C. Stochastic model checking of genetic circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* **2014**, *11*, 23.

(48) Nguyen, T.; Roehner, N.; Zundel, Z.; Myers, C. J. A Converter from the Systems Biology Markup Language to the Synthetic Biology Open Language. *ACS Synthetic Biology* **2016**, *5*, 479–486, PMID: 26696234.

(49) Roehner, N.; Myers, C. J. A methodology to annotate Systems Biology Markup Language Models with the Synthetic Biology Open Language. *ACS Synth. Biol.* **2014**, *3*, 57–66.

(50) Myers, C. J.; Barker, N.; Jones, K.; Kuwahara, H.; Madsen, C.; Nguyen, N.-P. D. iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics* **2009**, *25*, 2848–2849.

(51) Madsen, C.; Myers, C.; Patterson, T.; Roehner, N.; Stevens, J.; Winstead, C. Design and test of genetic circuits using iBioSim. *IEEE Design and Test* **2012**, 32–39.

(52) Sauro, H. M.; Choi, K.; Medley, J. K.; Cannistra, C.; Konig, M.; Smith, L.; Stocking, K. Tellurium: A Python Based Modeling and Reproducibility Platform for Systems Biology. *bioRxiv* **2016**, 054601.

(53) Chandran, D.; Sauro, H. M. Hierarchical modeling for synthetic biology. *ACS synthetic biology* **2012**, *1*, 353–364.

(54) Demir, E. et al. The BioPAX community standard for pathway data sharing. *Nat Biotech* **2010**, *28*, 935–942.

(55) Hedley, W. J.; Nelson, M. R.; Bellivant, D. P.; Nielsen, P. F. A short introduction to CellML. *Philos. T. Roy. Soc. A* **2001**, *359*, 1073–1089.

(56) Waltemath, D.; Adams, R.; Bergmann, F.; Hucka, M.; Kolpakov, F.; Miller, A.; Moraru, I.; Nickerson, D.; Sahle, S.; Snoep, J.; Le Novere, N. Reproducible computational biology experiments with SED-ML - The Simulation Experiment Description Markup Language. *BMC Systems Biology* **2011**, *5*, 198+.

(57) Bergmann, F. T.; Rodriguez, N.; Le Novère, N. COMBINE archive specification version 1. *Journal of Integrative Bioinformatics (JIB)* **2015**, *12*, 104–118.

(58) Eilbeck, K.; Lewis, S. E.; Mungall, C. J.; Yandell, M.; Stein, L.; Durbin, R.; Ashburner, M. The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology* **2005**, *6*, R44.

(59) Courtot, M. et al. Controlled vocabularies and semantics in systems biology. *Mol. Syst. Biol.* **2011**, *7*, 543.

(60) Juty, N.; Le Novere, N.; Laibe, C. Identifiers. org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic acids research* **2012**, *40*, D580–D586.

(61) Nielsen, A.; Der, B.; Shin, J.; Vaidyanathan, P.; Paralanov, V.; Strychalski, E.; Ross, D.; Densmore, D.; Voigt, C. Genetic circuit design automation. *Science* **2016**, *352*, aac7341.

(62) Der, B. S.; Glassey, E.; Bartley, B. A.; Enghuus, C.; Goodman, D. B.; Gordon, D. B.; Voigt, C. A.; Gorochowski, T. E. DNAplotlib: Programmable Visualization of Genetic Designs and Associated Data. *ACS Synthetic Biology* **0**, *0*, null.

(63) Czar, M. J.; Cai, Y.; Peccoud, J. Writing DNA with GenoCAD. *Nucleic Acids Research* **2009**, *37*, W40.

(64) Myers, C. J.; Barker, N.; Jones, K.; Kuwahara, H.; Madsen, C.; Nguyen, N.-P. D. iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics* **2009**, *25*, 2848.

(65) Hillson, N. J.; Rosengarten, R. D.; Keasling, J. D. j5 DNA assembly design automation software. *ACS Synth. Biol.* **2012**, *1*, 14–21.

(66) Misirli, G.; Hallinan, J.; Yu, T.; Lawson, J.; Wimalaratne, S.; Cooling, M.; Wipat, A. Model annotation for synthetic biology: automating model to nucleotide sequence conversion. *Bioinformatics* **2011**, *27*, 973–979.

(67) Bhatia, S.; Densmore, D. Pigeon: a design visualizer for synthetic biology. *ACS Synth. Biol.* **2013**, *2*, 348–350.

(68) Woodruff, L. B.; Gorochowski, T. E.; Roehner, N.; Mikkelsen, T. S.; Densmore, D.;

Gordon, D. B.; Nicol, R.; Voigt, C. A. Registry in a tube: multiplexed pools of retrievable parts for genetic design space exploration. *Nucleic Acids Research* **2016**, gkw1226.

(69) Pocock, M.; Taylor, C.; McLaughlin, J.; Misirli, G.; Wipat, A. An Environment for Augmented Biodesign Using Integrated Data Resources. 8th International Workshop on Bio-Design Automation. 2016.

(70) McLaughlin, J.; Pocock, M.; Misirli, G.; Madsen, J.; Wipat, A. VisBOL: Web-Based Tools for Synthetic Biology Design Visualization. *ACS Synthetic Biology* **2016**, *5*, 874–876.

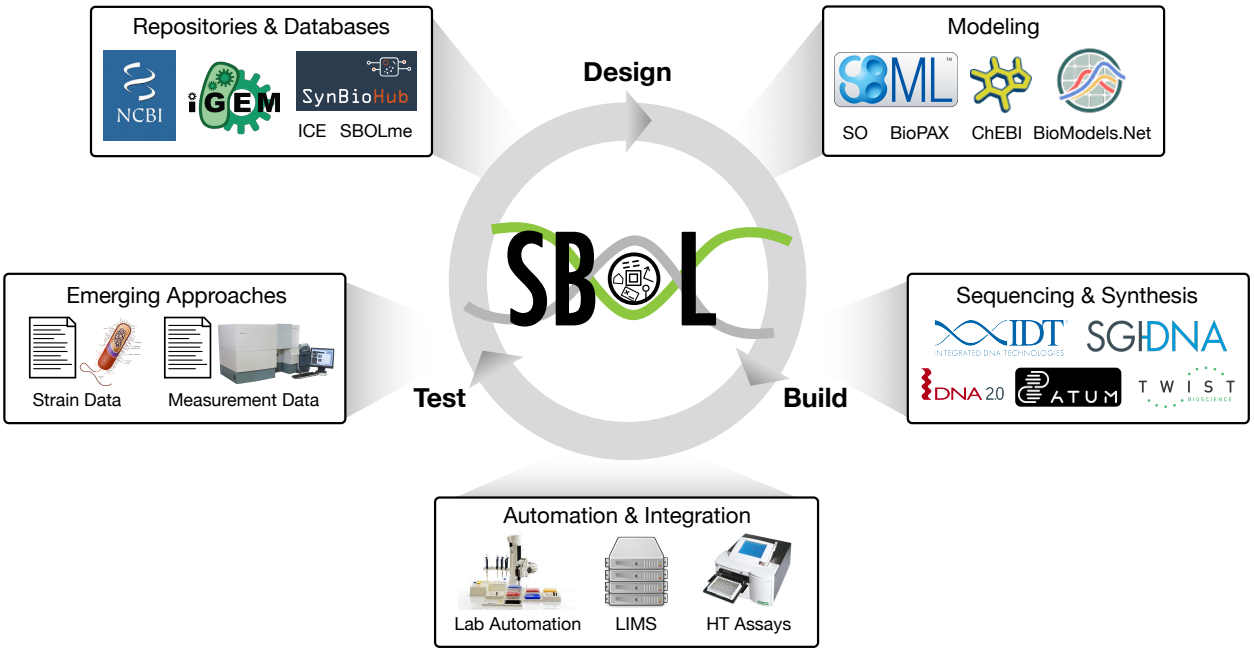

Figure 1: SBOL provides a shared representation for flexibly constructing workflows that may involve many different types of biological engineering resources, tools, and processes.

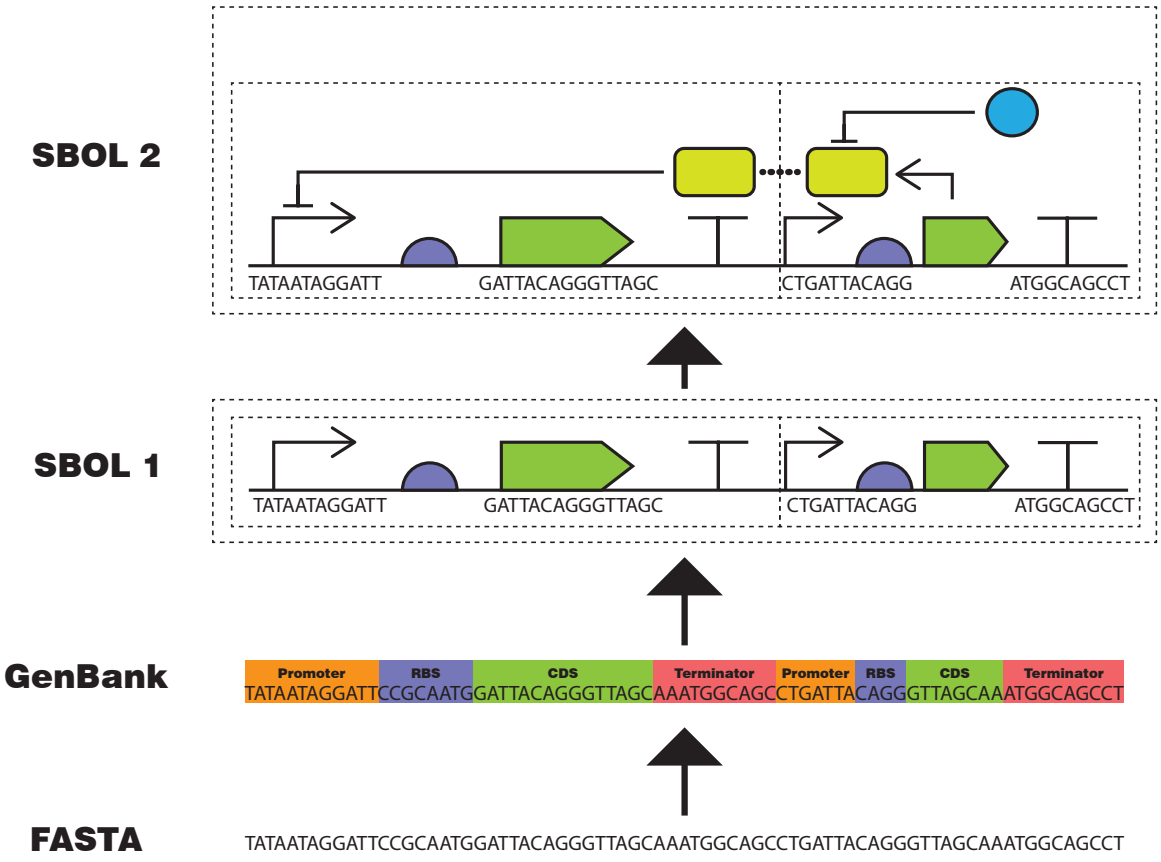

Figure 2: SBOL extends beyond prior sequence-centric formats like FASTA and GenBank to enable modular, hierarchical representations of both structure and function of a genetic design. SBOL 1 enables hierarchical composition of DNA components, some perhaps without sequences assigned, while SBOL 2 allows for more types of components and their interactions to be expressed (figure courtesy of Zundel et al.[29]).

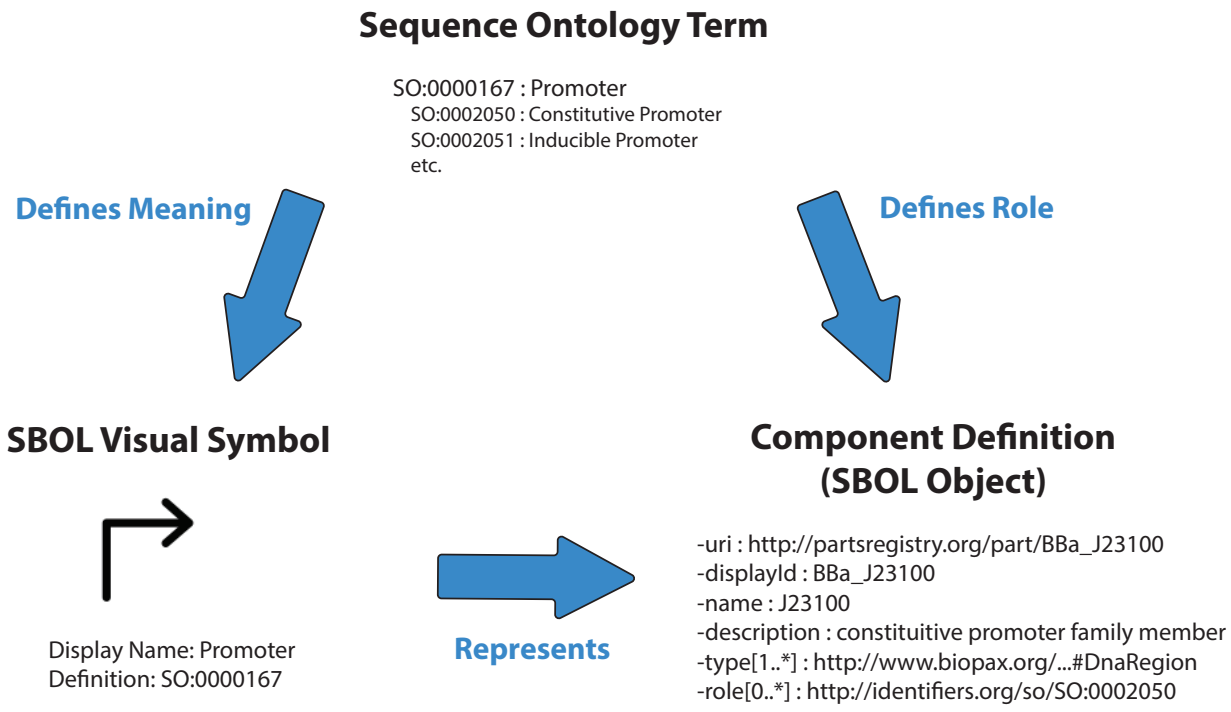

Figure 3: SBOLv is linked to the SBOL data model by shared use of SO terms (figure courtesy of Zhang et al.[36])

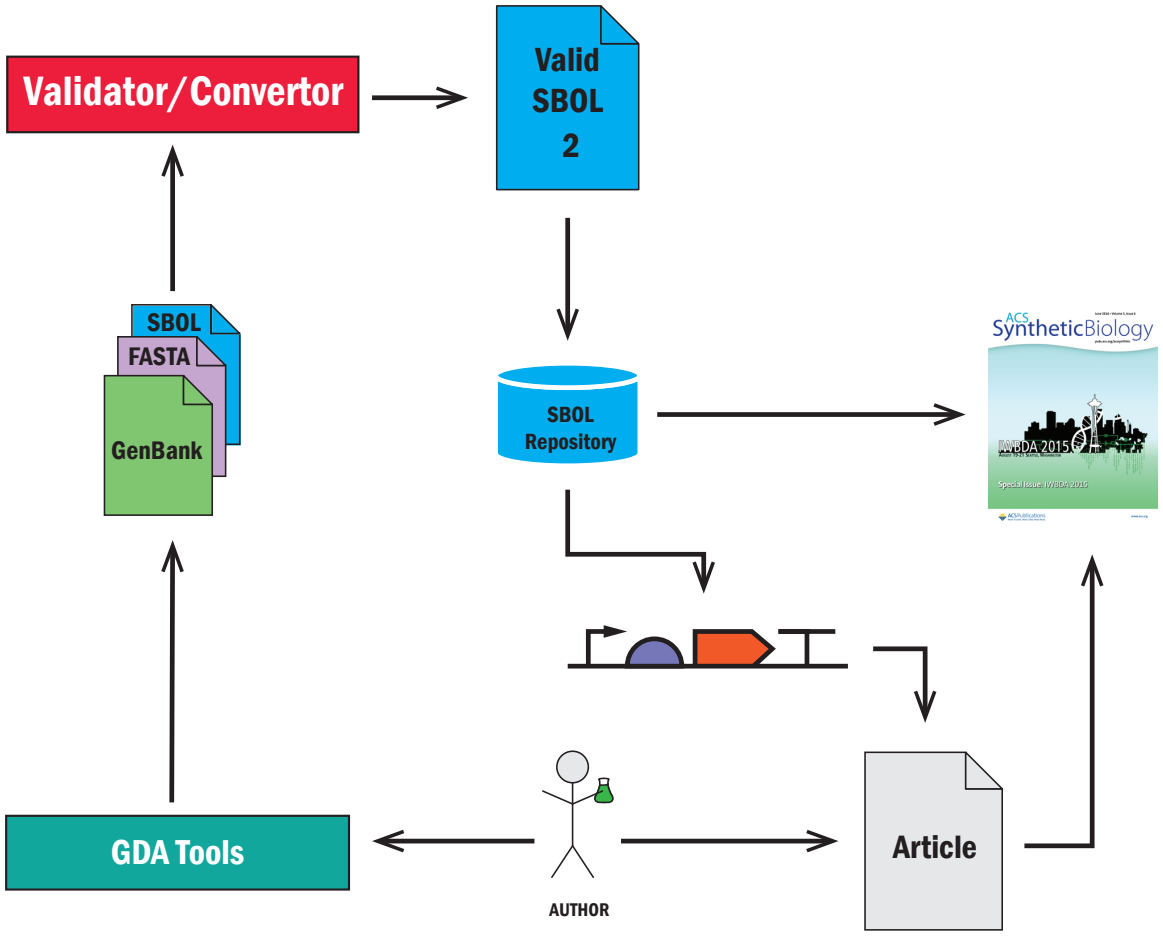

Figure 4: ACS Synthetic Biology workflow for integration of published articles with machine-readable SBOL representations of the biological constructs described by those articles. The author constructs their design using the genetic design automation (GDA) tool(s) of their choice producing a description in FASTA, GenBank, or preferably SBOL. Their design is then converted into a valid SBOL 2 document that is deposited in an SBOL repository. A link to this data and potentially an SBOLv diagrarm are published with the article (figure courtesy of Zundel et al.[29]).

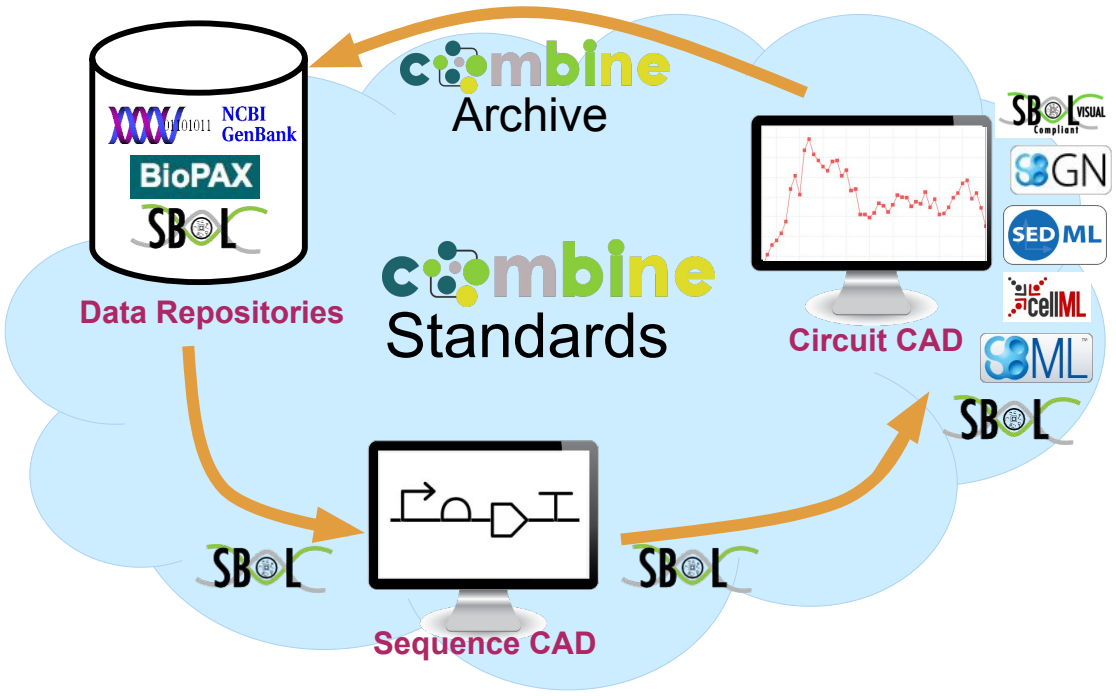

Figure 5: A standard-Enabled Workflow for Synthetic Biology using COMBINE standards. The use of standards provides rich data repositories, consistent annotations, lossless data conversions, intuitive visualizations, and seamless connections of tools.